Background Processes

Background processing can be implemented in situations where processing requirements could cause significant response time problems during dialog operation.

This is usually the case if a large number of database accesses or other time consuming processing is necessary.

Typical examples of this type of processing are complex deletion procedures, complex calculations or modifications, which require access to a large number of data records.

Depending on specific requirements, background processes can either be started from an online dialog or from the computer center.

The background program frame handles parameter transfer from the dialog system, error correction as well as components to restart background processes in the event of an abnormal termination.

The following functions are provided for the end user:

- online monitoring of background program execution;
- online display of errors detected during background processing;
- restart background processing following abnormal termination.

Background processes which are not implemented with the appropriate background frames cannot use the features described above.

The following topics are covered below:

- General Information
- Creating and Maintaining Background Procedures
- Invoking Background Processes
- Start Background Program from Dialog
- Implementing Background Programs
- Implementing Computer Center Background Processes

General Information

Background procedures must be defined in the application shell for the background process to which the background program belongs.

To invoke the background process, the subprogram ZXBG010N is called by the online program. For further information see the corresponding procedure description.

This subprogram ensures that the parameters are transferred using a control record in a control file. This control record is identified by a unique key (the timestamp).

The timestamp is passed as a standard parameter to the batch program.

The background process is then submitted to the operating system via a standard interface. The background procedure is therewith submitted to the operating system from the dialog program. The background program is then started.

The writing of a control record in the file Z_BG_PARM as well as the starting of the background process are executed from the subprogram ZXBG010N.

Therefore, it is necessary to ensure that the parameters are correctly set for the call and that the call to the subprogram ZXBG010N is coded.

Creating and Maintaining Background Procedures

General

For each background process, operating-system-dependent background procedures are required.

Note:
Background procedures must be created prior to implementing background programs. The parameters for the background procedures are required during the dialog call to the background process.

Using Administration Functions

The application shell is used to define background procedures.

Use the functions Add Background Procedure, Modify Background Procedure, etc., to create and maintain the background procedure.

For a full description, see the Natural Application Shell documentation.

Types of Background Processing

A background procedure is not needed for each background process. Instead, the various types of background processing should be identified and a background procedure for each corresponding type is created.

The following are examples of various types of background processes:

- receipt of data from another system;
- transfer of data to another system;
- copying/deleting/modifying mass volumes of data;
- printing of lists;
- loading/unloading data.

Invoking Background Processes

Calling a Background Program from a Dialog

A background program can be started from any dialog.

To call the background program from a dialog

- Enter CALLNAT ’ZXBG010N’ USING PZ_BG_START_BATCH.
  This CALLNAT contains END TRANSACTION and BACKOUT TRANSACTION statements.

Note:
When writing programs which update data, you must ensure that the procedure is called at a point at which it will not adversely affect the transaction logic of the dialog.

Parameter Usage

Both the background program and the background procedure needed for the execution of the background program require certain parameters. These parameters must be passed by the dialog whenever the subprogram ZXBG010N is invoked.

The required parameters can be displayed with the application shell function Display Background Procedure.

The parameters for the background program, for example, selection/sort criteria, which are assigned values in the online program, must also be passed.

Detailed information on these parameters is contained in the procedure description.

Start Background Program from Dialog

The subprogram ZXBG010N ensures that at runtime the parameters are transferred to the control file, from which they can be read by the background program.

Natural Object Name:	ZXBG010N
Parameter:	ZXBG010A (PDA)

Parameters

Group PZ_BG_START_BATCH of the PDA ZXBG010A.

Input/Output	Parameter Variable	Description
Input	PZ_BG_PTS	
Output	PZ_BG_RSP	0 - Ok 1 - Background procedure not found 2 - Background parameter not found (restart) 3 - Background parameter not stored 9 - Error during submit call
Output	PZ_BG_MSG_NUM	
Output	PZ_BG_MSG_FILL	
Input	PZ_BG_DELIM	Input delimiter
Input	PZ_BG_CLIENT_ID	Client ID
Input	PZ_BG_US_ID	User ID
Input	PZ_BG_PSW	Password
Input	PZ_BG_NAME	Background process name or title (if applicable for the operating system)
Input	PZ_BG_LIB	Library in which program will run
Input	PZ_BG_PGM	Program name to be executed
Input	PZ_BG_NATPARM	Natural parameter module
Input	PZ_BG_PRIORITY	Priority of the background procedure (if applicable for the operating system)
Input	PZ_BG_ONLINE_LIB	Library from which the start of the background process is invoked
Input	PZ_BG_ONLINE_PGM	Program to be called to start the background process
Input	PZ_BG_FU_ID	Function ID
Input	PZ_BG_DESCR_LC	Description
Input	PZ_BG_LA_ID	Language ID
Input	PZ_BG_LA_NAT_CODE	Natural language code
Input	PZ_BG_CD_ID	ID of the background procedure
Input	PZ_BG_PRINTER (1:5)	Printer name
Input	PZ_BG_WORKFILE (1:5)	Work file path and name
Input	PZ_BG_SUST_NAME?(1:5)	Name of substitution variable
Input	PZ_BG_SUBST_VALUES (1:5)	Content of substitution variable
Input	PZ_BG_RESTART	Restart indicator
Input	PZ_BG_PGM_PARM (1:5)	Parameter to be passed to the background program

Implementing Background Programs

The frame gallery provides three application frames for background processing:

- Background program.
- Load objects.
- Unload objects

These are described in section Application Frames.

This section provides additional information on the use of the background program frame.

- Passing Parameters
- Logically Locking Data Records
- Restart
- Error Handling
- Setting the Processing Status
- Monitoring Program Execution

Passing Parameters

The background program receives a timestamp as an input parameter.

This timestamp is used to read the corresponding control record from the control file Z_BG_PARM. This is a part of the frame functionality.

The parameters from the online program, for example, selection criteria, are made available. The transfer of parameters is thereby automatic.

The parameters are available to the background program via the variable LZ_PGM_PARM.

Logically Locking Data Records

Data records can be logically locked by background programs.

This may be necessary, for example, if the data must be modified by the background program, and at the same time processed by the dialog system.

Locking can also be required for read access. For example, a statistic which is based on certain data values requires that these data records remain unchanged during execution of the calculation of the statistic.

Locking Data Records

Use the inline subroutine Z_CHECK_AND_LOCK_RECORD to lock individual data records, data areas or objects, as required within a business application.

After execution of this subroutine, an END TRANSACTION must follow. The program must include the necessary processing logic.

The parameters are contained in the LDA ZXFBA00L.

Input/Output	Parameter Variable	Description
Input	LZ_LOCK_OBJ_ID	Identifier of the object type
Input	LZ_LOCK_KEY	The key of the record to be locked, or the beginning of the record range to be locked.
Input	LZ_LOCK_KEY_END	The end of the record area to be locked.
Output	LZ_VAL_ERR	TRUE: Locking is not possible.

If LZ_VAL_ERR has a setting of TRUE, the subroutine also inserts an error number into the variable LZ_MSG_NUM(1).

Example

```
MOVE  'ARTICLE'        TO LZ_LOCK_OBJ_ID
MOVE  PZ_LOCAL.PZ_KEY  TO LZ_LOCK_KEY

PERFORM Z_CHECK_AND_LOCK_RECORD

IF LZ_VAL_ERR
  BACKOUT TRANSACTION
  PERFORM Z_TERMINATE_PROCESS
ELSE
  END TRANSACTION
END-IF
```

Releasing Data Records

The background program frame does not contain any locking logic. Therefore, if you use the procedure for locking data records, you must also release the records following completion of processing.

Use the inline subroutine Z_CANCEL_LOCK_RECORD for this purpose.

After execution of this subroutine, an END TRANSACTION must follow. The program must include the necessary processing logic.

Restart

A background program can be restarted in the event it is terminated abnormally, provided that the portion of the processing which has been successfully completed must not be repeated.

Restart logic is recommended when:

- processing involving data modifications is to be executed, or
- extensive list processing is to be executed.

As a prerequisite for implementing restart logic, the restart points must be logged at runtime.

This entry is written immediately prior to execution of an END TRANSACTION statement and is confirmed together with the data modifications via the END TRANSACTION statement.

Use the inline subroutine Z_STORE_RESTART_DATA to write the restart data.

The call is contained in the background program frame. The data are written following logical transactions to the file Z_BG_PARM.

The number of transactions following which an END TRANSACTION statement is to be executed can be set using the variable C#TRANSACTION. The default value for this variable is 99.

Before calling this procedure, the desired restart point, for example, the key value of a database record, must be provided in LZ_RESTART_DATA(*).

The restart of the abnormally terminated background program is then performed via the application shell. For further information, see the Natural Application Shell documentation.

With this function you can view/modify existing restart data, and then restart the background process.

The frames of the batch programs ensure that the restart data provided in the variable LZ_RESTART_DATA(*) are received and are available for restart processing.

Error Handling

Errors which occur during background program execution can be logged. Differentiation between the following types of errors must be made:

- Natural runtime errors
- Application errors

Natural Runtime Errors

Natural runtime errors can be detected in the background program with the ON ERROR statement and automatically displayed in the application shell ’’Maintain Error Log" dialog.

All database modifications which were executed during the current, not yet successfully closed logical transaction, are backed out of the database and the corresponding logical locks are released. This is a part of the frame functionality.

Application Errors

Application errors can be handled in various ways:

- termination of the background program with an entry in the error log file. The inline subroutine Z_TERMINATE_PROCESS can be used for this purpose. Error handling is the same as that for Natural runtime errors.
- continuation of the background program with an entry (warning) in the error log file. The inline subroutine Z_STORE_ERROR_LOG can be used for this purpose. After execution, an END TRANSACTION must follow. In this case, the program can be closed with the status ’Process has ended with an error/warning’. This is performed by the frames when the variable LZ_APPL_ERR is set to TRUE.

In each case, these subroutines must be supplied with the variables ZER_USER_DESC(1:4) and ZER_APPL_ERR_NUM.

Display Error Log

Error logs can be displayed using the application shell function Browse Error Log.

For further information, see the Natural Application Shell documentation.

Setting the Processing Status

A background program is assigned a status when it is started from a dialog. This status can be displayed via the application shell function Browse Background Process.

Terminating a Background Program

Normally the appropriate end status is set whenever the background program is ended. This status can however be directly set to: ’Process ended with an error/warning’ by setting the variable LZ_APPL_ERR to TRUE.

Termination by Calling another Background Program

If a background program was called from another background program, and control is to be returned to the calling program, the background status may not be modified.

This can be done by setting the variable LZ_CONTINUE to TRUE.

The last program of the background process will set the status to ENDED.

Monitoring Program Execution

The application shell provides various functions for monitoring background processes. The following information is provided:

Function	Information
Browse Background Process	Status of user’s background processes
Browse Error Log	Display errors

No access is available to the operating system itself, i.e., direct intervention with executing background processes is not possible.

Implementing Computer Center Background Processes

No frame is available to implement background programs which are to be started by computer center personnel.

Note:
The background program is not intended for this purpose. It is intended only for the implementation of batch programs which are to be started from a dialog program.

Error Handling

Natural runtime errors should be handled using the ON ERROR statement.

Monitoring Program Execution

The monitoring of batch jobs (status, restart) is not supported by the application shell.

